دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

کارشناسی ارشد علوم کامپیوتر گرایش داده کاوی

پروژه شماره پنج درس داده کاوی

نگارش

حدیث حق‌شناس جزی

استاد راهنما

مهدی قطعی

استاد مشاور

بهنام یوسفی مهر

آذر ۱۴۰۱

چکیده

در این گزارش هدف این است که با استفاده از دیتای تصادفات آمریکا در سال های ۲۰۱۶ تا ۲۰۲۱ ، که در سراسر این کشور جمع آوری شده است و شامل ۴۹ ایالت ایالات متحده میشود، داده های تصادفات را طبقه بندی کنیم. API های جمع آوری شده در این دیتاست توسط نهاد های مختلف مانند وزارت حمل و نقل و حسگر های ترافیک در شبکه های جاده ای گرفته شده است و شامل حدود ۲,۸ میلیون تصادف میشود که شدت های متفاوتی دارند و در شرایط متفاوتی اتفاق افتاده اند. در ادامه به بررسی و پیش پردازش دیتاست و اعمال روش های مختلف طبقه بندی مانند درخت تصمیم و بیز گوسین و شبکه عصبی و تغییر هایپر پارامتر های این مدل ها میپردازیم و نتایج بدست آمده را با یکدیگر مقایسه میکنیم.

فصل اول

مقدمه

مقدمه

به منظور مقایسه روش های طبقه بندی ذکر شده در چکیده گزارش، ابتدا با کمک کد های متفاوت به بررسی ویژگی های دیتاست و حذف ستون هایی با میزاین مسیسینگ دیتای بالا و یا جایگزین کردن آنها میپردازیم. در این دیتاست چند مدل داده متفاوت وجود دارد که به کمک encoder ویژگی های توصیفی را به داده های ۰-۱ ای تبدیل میکنیم. سپس یک به یک الگوریتم درخت تصمیم و دیگر الگوریتم هارا بر روی داده های آموزشی و آزمایشی اعمال کرده و به نتیجه گیری از روی میزان دقت و بایاس خواهیم پرداخت.

فصل دوم

پیش پردازش داده ها

۲-۱ پیش پردازش دیتاست

در ابتدا فایل دیتا را فراخوانی میکنیم. داده شامل تقریبا ۲ میلیون و ۸۰۰ هزار تصادف در ۳۶ ستون میباشد.

اطلاعات کلی دیتاست در تصویر زیر قابل مشاهده است:

```
In [2]: import pandas as pd
        import numpy as np
        import os

In [3]: df = pd.read_csv("US_Accidents_Dec21_updated.csv")

In [57]: df.head()
```

Out[57]:

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	...	Roundabout	Station	Stop	Traffic_Calmin
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230	Between Sawmill Rd/Exit 20 and OH-315/Olentang...	...	False	False	False	Fals
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747	At OH-4/OH-235/Exit 41 - Accident.	...	False	False	False	Fals
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055	At I-71/US-50/Exit 1 - Accident.	...	False	False	False	Fals
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123	At Dart Ave/Exit 21 - Accident.	...	False	False	False	Fals
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500	At Mitchell Ave/Exit 6 - Accident.	...	False	False	False	Fals

5 rows × 36 columns

انواع داده را در تصویر زیر میتوانیم مشاهده کنیم که شامل انواع مختلف داده های توصیفی و عددی میباشد.

```
In [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ID                     object
 1   Severity               int64
 2   Start_Time             object
 3   End_Time               object
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object
 10  Number                 float64
 11  Street                 object
 12  Side                   object
 13  City                   object
 14  County                 object
 15  State                  object
 16  Zipcode                object
 17  Country                object
 18  Timezone               object
 19  Airport_Code           object
 20  Weather_Timestamp      object
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object
 30  Amenity                bool
 31  Bump                   bool
 32  Crossing               bool
 33  Give_Way               bool
 34  Junction               bool
 35  No_Exit                bool
 36  Railway                bool
 37  Roundabout             bool
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 773.4+ MB
```

میزان دیتای های خالی در تصویر زیر مشاهده میشود که در بعضی ستون ها بسیار زیاد میباشند لذا آنهارا حذف میکنیم.

```
In [8]: df.isnull().sum()
```

```
Out[8]: ID                          0
        Severity                    0
        Start_Time                  0
        End_Time                    0
        Start_Lat                   0
        Start_Lng                   0
        End_Lat                     0
        End_Lng                     0
        Distance(mi)                0
        Description                 0
        Number                1743911
        Street                      2
        Side                        0
        City                      137
        County                      0
        State                       0
        Zipcode                  1319
        Country                     0
        Timezone                 3659
        Airport_Code             9549
        Weather_Timestamp       50736
        Temperature(F)          69274
        Wind_Chill(F)          469643
        Humidity(%)             73092
        Pressure(in)            59200
        Visibility(mi)          70546
        Wind_Direction          73775
        Wind_Speed(mph)        157944
        Precipitation(in)      549458
        Weather_Condition       70636
        Amenity                     0
        Bump                        0
        Crossing                    0
        Give_Way                    0
        Junction                    0
        No_Exit                     0
        Railway                     0
        Roundabout                  0
        Station                     0
        Stop                        0
        Traffic_Calming             0
        Traffic_Signal              0
        Turning_Loop                0
        Sunrise_Sunset           2867
        Civil_Twilight           2867
        Nautical_Twilight        2867
        Astronomical_Twilight    2867
        dtype: int64
```

```
In [9]: df.drop('Number', axis=1, inplace=True)
```

```
In [10]: df.drop('Weather_Timestamp', axis=1, inplace=True)
         df.drop('Temperature(F)', axis=1, inplace=True)
         df.drop('Wind_Chill(F)', axis=1, inplace=True)
         df.drop('Humidity(%)', axis=1, inplace=True)
         df.drop('Pressure(in)', axis=1, inplace=True)
         df.drop('Visibility(mi)', axis=1, inplace=True)
         df.drop('Wind_Direction', axis=1, inplace=True)
         df.drop('Wind_Speed(mph)', axis=1, inplace=True)
         df.drop('Precipitation(in)', axis=1, inplace=True)
         df.drop('Weather_Condition', axis=1, inplace=True)
```

سپس اقدام به تبدیل داده های توصیفی به عددی میکنیم.

In [21]:
```
import category_encoders as ce

encoder = ce.OrdinalEncoder(cols=obj_df.columns)

X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
```

In [22]:
```
X_train.head()
```

Out[22]:

	ID	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Street	...	Roundabout	Station	Stop
584031	1	1	1	32.664252	-97.242618	32.668152	-97.234828	0.527	1	1	...	False	False	False
1393597	2	2	2	34.055514	-118.192618	34.054717	-118.202428	0.564	2	2	...	False	False	False
1013445	3	3	3	36.982306	-121.864820	36.985193	-121.865405	0.202	3	3	...	False	False	False
2728434	4	4	4	38.635500	-90.410470	38.635590	-90.400710	0.527	4	4	...	False	False	False
865988	5	5	5	37.354209	-120.629522	37.345492	-120.615976	0.957	5	5	...	False	False	False

5 rows × 35 columns

فصل سوم

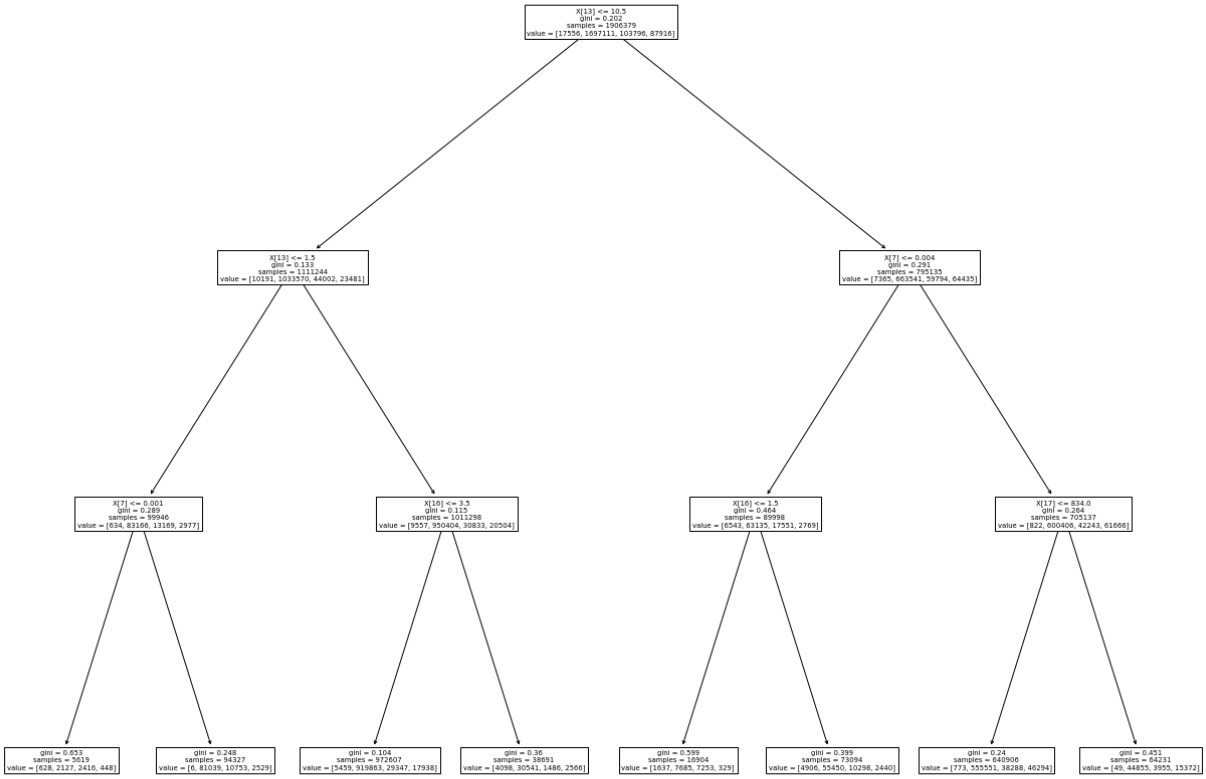
پیاده سازی و مقایسه الگوریتم ها

۳-۱ الگوریتم درخت تصمیم

در زیر الگوریتم درخت تصمیم پیاده سازی شده است :

```
In [39]: from sklearn.tree import DecisionTreeClassifier
         dt= DecisionTreeClassifier()
         dt.fit(X_train,y_train)
         print("score :",dt.score(X_test,y_test))

         score : 0.6146706526242248

In [40]: from sklearn.tree import DecisionTreeClassifier
         dt= DecisionTreeClassifier(max_depth=8)
         dt.fit(X_train,y_train)
         print("score :",dt.score(X_test,y_test))

         score : 0.8959788617868861

In [41]: from sklearn.tree import DecisionTreeClassifier
         dt= DecisionTreeClassifier(max_depth=16)
         dt.fit(X_train,y_train)
         print("score :",dt.score(X_test,y_test))

         score : 0.834675061743647

In [42]: from sklearn.tree import DecisionTreeClassifier
         dt= DecisionTreeClassifier(max_depth=32)
         dt.fit(X_train,y_train)
         print("score :",dt.score(X_test,y_test))

         score : 0.6298203443586169
```

همانطور که مشاهده میشود با اضافه کردن عمق درخت تا عدد ۸ نتیجه بهتر میشود اما سپس کاهش پیدا میکند.



در شکل بالا نیز درخت تصمیم با عمق ۴ را میتوانیم تماشا کنیم.

۳-۲ الگوریتم بیز

در اینجا به کمک کتابخانه sklearn الگوریتم گوسی را پیاده سازی میکنیم . که تقریبا عدد بهتری از درخت تصمیم مخصوصا با تقلیل واریانس هموار سازی به ما میدهد:

```
In [45]: from sklearn.naive_bayes import GaussianNB

In [48]: gb = GaussianNB()
         gb.fit(X_test, y_test)

Out[48]: GaussianNB()

In [51]: print("Naive Bayes score: ",gb.score(X_test, y_test))

         Naive Bayes score:  0.8383759530460732

In [53]: gb = GaussianNB(var_smoothing = 1e-06)
         gb.fit(X_test, y_test)
         print("Naive Bayes score: ",gb.score(X_test, y_test))

         Naive Bayes score:  0.8585940021065793

In [54]: gb = GaussianNB(var_smoothing = 1e-03)
         gb.fit(X_test, y_test)
         print("Naive Bayes score: ",gb.score(X_test, y_test))

         Naive Bayes score:  0.8718234903824751

In [55]: gb = GaussianNB(var_smoothing = 1e-01)
         gb.fit(X_test, y_test)
         print("Naive Bayes score: ",gb.score(X_test, y_test))

         Naive Bayes score:  0.8895185433291833
```

همانطور که مشاهده میشود هرچه هموار سازی داده را از عدد ۹ به ۱ تقلیل داده ایم نتایج بهتری را در این الگوریتم مشاهده میکنیم.

۲-۳ الگوریتم شبکه عصبی

همانطور که در ۳ تصویر زیر مشاهده میشود شکل کلی شبکه عصبی ساخته شده و سپس دقت و خطا در ۱۰ اپاک مشخص شده است. دقت در این مدل از هر دوی قبلی بیشتر است.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 28)                812

 dense_1 (Dense)             (None, 20)                580

 dense_2 (Dense)             (None, 4)                 84

=================================================================
Total params: 1,476
Trainable params: 1,476
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/10
6165/6165 [==============================] - 8s 1ms/step - loss: 0.4549 - categorical_accuracy: 0.8876 - val_loss: 0.4464 -
val_categorical_accuracy: 0.8905
Epoch 2/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4472 - categorical_accuracy: 0.8901 - val_loss: 0.4458 -
val_categorical_accuracy: 0.8905
Epoch 3/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4469 - categorical_accuracy: 0.8901 - val_loss: 0.4456 -
val_categorical_accuracy: 0.8905
Epoch 4/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4467 - categorical_accuracy: 0.8901 - val_loss: 0.4455 -
val_categorical_accuracy: 0.8905
Epoch 5/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4466 - categorical_accuracy: 0.8901 - val_loss: 0.4453 -
val_categorical_accuracy: 0.8905
Epoch 6/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4465 - categorical_accuracy: 0.8901 - val_loss: 0.4452 -
val_categorical_accuracy: 0.8905
Epoch 7/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4463 - categorical_accuracy: 0.8901 - val_loss: 0.4451 -
val_categorical_accuracy: 0.8905
Epoch 8/10
6165/6165 [==============================] - 8s 1ms/step - loss: 0.4462 - categorical_accuracy: 0.8901 - val_loss: 0.4449 -
val_categorical_accuracy: 0.8905
Epoch 9/10
6165/6165 [==============================] - 8s 1ms/step - loss: 0.4461 - categorical_accuracy: 0.8901 - val_loss: 0.4448 -
val_categorical_accuracy: 0.8905
Epoch 10/10
6165/6165 [==============================] - 7s 1ms/step - loss: 0.4460 - categorical_accuracy: 0.8901 - val_loss: 0.4447 -
val_categorical_accuracy: 0.8905

<keras.callbacks.History at 0x1aaa3de4c70>
```

```
0.8904907267012763
```

جمع بندی

طبق مشاهدات از ۳ الگوریتم که در طی گزارش دیدیم دقت در مواردی در درخت تصمیم بهبود یافت اما در مدل بیز به طور کلی دقت بهتری داشتیم و نهایتا در شبکه عصبی به بهترین دقت و کمترین خطا میرسیم.

منابع و مراجع:

- سایت kaggle.com

پایان